CONTENTS

No.	Date	Name of the Experiment	Page No	Marks awarded	Signature of the staff
1	07/07/2020	R-CONSOLE and Data Analysis using R	3		
2	16/07/2020	R Expression and data structure	9		
3	28/07/2020	Implement Classification using K nearest Neighbour Classification	15		
4	6/08/2020	Implement Classification using Decision Classification	21		
5	11/08/2020	Implement Classification using Random Forest Classification	27		
6	27/08/2020	Implement Decision Tree Based Algorithm for Classification	32		
7	08/09/2020	Back Propagation Neural Network Algorithm	38		
8	17/09/2020	Implement K-means Algorithm for Clusters	43		
9	06/10/2020	Implement Hierarchical Clustering Algorithm	48		
10	22/10/2020	Implement Apriori Algorithm for Associate Rule	52		

EXERCISE NO:1 **DATA ANALYTICS USING PYTHON**

DATE:07/07/2020

AIM:

To implement data analytics using python.

DESCRIPTION:

Python has a unique attribute and easy to use when it comes to quantitative and analytical computing.

Time spent on debugging codes and on various software engineering constraints are also minimized.

The time for code implementation is less which has developed and software engineers to spend more time to work in their algorithms.

Python provides a massive database of libraries and artificial intelligence and machine learning.

1.Finding Factorial of a number

```
#take input from the user
num = as.integer(readline(prompt="Enter a number: "))
factorial = 1
#check is the number is negative, positive or zero
if(num<0)
{
  print("Sorry, factorial does not exist for negative numbers")
}
else if(num==0)
{
   print("The factorial of 0 is 1")
}
else
{
   for(i in 1:num)
   {
      factorial=factorial*i
   }
   print(paste("The factorial of",num,"is",factorial))
}
```

OUTPUT:

Enter a number: 8

[1] “The factorial of 6 is 720”

2.To check Prime Number

```
# Program to check if the input number is prime or not
# take input from the user
num=as.integer(readline(prompt="Enter a number: "))
flag=0
if(num>1)
{
  flag=1
  for(i in 2:(num-1))
  {
    if((num%%i)==0)
    {
      flag=0
      break
    }
  }
}
if(num==2)
  flag=1

if(flag==1)
{
  print(paste(num,"is a prime number"))
}
else
{
  print(paste(num,"is not a prime number"))
}
```

OUTPUT:

Enter a number: 25

[1] “15 is not a prime number”

Enter a number: 13

[1] “13 is a prime number”

Write a R program to get the length of the first two vectors of a given list.

```
list1=list(g1 = 1:10,g2 = "R Programming",g3 = "HTML")
print("Original list:")
print(list1)
print("Length of the vector ‘g1’ and ‘g2’ of the said list")
print(length(list1$g1))
print(length(list1$g2))
```

OUTPUT:

```
[1] "Original list:"
$g1
 [1]  1  2  3  4  5  6  7  8  9 10

$g2
[1] "R Programming"

$g3
[1] "HTML"

[1] "Length of the vector ‘g1’ and ‘g2’ of the said list"
[1] 10
[1] 1
```

RESULT:

Thus the data analytics using python are implemented.

EXERCISE NO:2 **DATA STRUCTURES USING PYTHON**

DATE:16/07/2020

AIM:

To implement data structures using python.

DESCRIPTION:

Python gives you some powerful, highly optimised data structures, both as built in and as part of a few modules in the standard library.

Combination of data structures such as lists, dictionaries and tuples are generally sufficient to complement more richer data structures that are needed in real life programming.

The list is powerful, yet simple, collection mechanism that provides the programmer with widely variety of options.

Using python, we can store a collection of related data efficiency.

//2

1.Write a R program to create a list containing strings, numbers, vectors and a logical values.

```
list_data = list("Python","PHP",c(5,7,9,11),TRUE,125.17,75.83)
print("Data of the list:")
print(list_data)
```

OUTPUT:

```
[1] "Data of the list:"
[[1]]
[1] "Python"

[[2]]
[1] "PHP"

[[3]]
[1]  5  7  9 11

[[4]]
[1] TRUE

[[5]]
[1] 125.17

[[6]]
[1] 75.83
```

2. Write a R program to merge two given lists into one list

```
n1=list(1,2,3)
c1=list("Red","Green","Black")
print("Original lists:")
print(n1)
print(c1)
print("Merge the said lists:")
mlist = c(n1, c1)
print("New merged list:")
print(mlist)
```

OUTPUT:

```
 [1] "Original lists:"
 [[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

 [[1]]
[1] "Red"
```

```
[[2]]
[1] "Green"

[[3]]
[1] "Black"

 [1] "Merge the said lists:"
 [1] "New merged list:"
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] "Red"

[[5]]
[1] "Green"
```

```
[[6]]

[1] "Black"
```

RESULT:

Thus the data structures in python are implemented.

EXERCISE NO:3 IMPLEMENT CLASSIFICATION USING K-NEAREST NEIGHBOUR

DATE:28/07/2020 CLASSIFICATION

AIM:

To implement classification using k- nearest neighbour classification by using python.

DESCRIPTION:

Knn works based on minimum distance from the query instance to the training samples to determine the k- nearest neighbour.

After we gather k- nearest neighbour we take simple majority of these k- nearest neighbour to the prediction of the query instance.

The data for knn generation consist of several multivariate attribute name that will be used to classify.

It uses all of the data for training while classifying a new data point or instance.

KNN ALGORITHM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("ads.csv")
data.head()
data.isnull().any()
x=data.iloc[:,1:4].values
y=data.iloc[:,4:5].values
x[:5]
y[:5]
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
x[:,0] = lb.fit_transform(x[:,0])
x[:5]
from sklearn.model_selection import train_test_split as tts
x_train,x_test,y_train,y_test = tts(x, y, test_size = 0.1,random_state=0)
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier ( n_neighbors = 5 , p = 2 )
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
y_pred
y_test
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
from sklearn.metrics import confusion_matrix
import sklearn.metrics as metrics
fpr, tpr ,threshold = metrics.roc_curve(y_test,y_pred)
roc_auc = metrics.auc(fpr,tpr)
roc_auc
plt.plot(fpr,tpr)
 plt.xlim([0,1])
plt.ylim([0,1])
 plt.style.use("dark_background")
```

OUTPUT:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
User ID            False
Gender             False
Age                False
EstimatedSalary    False
Purchased          False
dtype: bool
```

```
array([['Male', 19, 19000],
['Male', 35, 20000],
['Female', 26, 43000],
['Female', 27, 57000],
['Male', 19, 76000]], dtype=object)
```

```
 array( [ [ 0 ],
          [ 0 ],
          [ 0 ],
          [ 0 ],
          [ 0 ] ], dtype=int64 )
```

```
  array( [ [ 1,   19,   19000 ] ,
           [ 1,   35,   20000 ] ,
           [ 0,   26,   43000 ] ,
           [ 0,   27,   57000 ] ,
           [ 1,   19,   76000 ] ] , dtype=object)
```

```
 KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=5, p=2, weights='uniform')
```

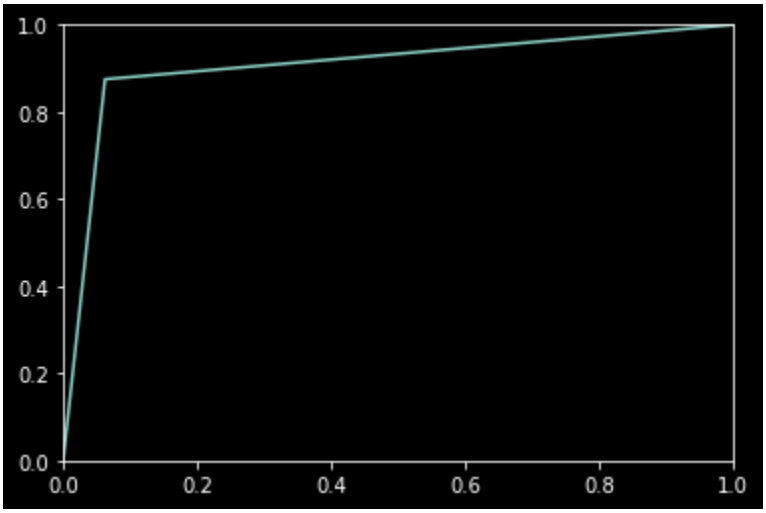
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
```

0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1], dtype=int64)

array([[0],
[0],
[0],
[0],
[0],
[0],
[1],
[0],
[0],
[0],
[0],
[0],
[0],
[0],
[0],
[0],
[0],
[0],
[1],
[0],
[0],
[1],
[0],
[1],
[0],
[1],
[0],
[0],
[0],
[0],
[0],
[1],
[1],
[0],
[0],
[0],
[0],
[0],
[0],
[0]], dtype=int64)

0.925

0.90625



RESULT:

Thus the classification using k-nearest neighbour classification using python is implemented.

EXERCISE NO:4 IMPLEMENT CLASSIFICATION USING DECISION

DATE:06/08/2020 CLASSIFICATION

AIM:

To implement classification using decision classification using python.

DESCRIPTION:

Decision tree algorithm works for both continuous as well as categorised output variables.

The best attribute is placed on the root node of the tree.

The training set of the data set is splitted into subsets.

While maintaining the subset make sure that each subset of training data set should have the same value for an attribute.

The leaf nodes in all branches are found.

Decision Tree

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("ads.csv")
data.head()
data.isnull().any()
x = data.iloc[:,1:4].values
y = data.iloc[:,4:5].values
x
y[:10]
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
x[:,0] = lb.fit_transform(x[:,0])
x
from sklearn.model_selection import train_test_split as tts
x_train,x_test,y_train,y_test = tts(x, y, test_size = 0.1,random_state=0)
x_train
x_test[:10]
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='entropy')
dt.fit(x_train,y_train)
y_pred = dt.predict(x_test)
y_pred
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
import sklearn.metrics as metrics
fpr, tpr ,threshold = metrics.roc_curve(y_test,y_pred)
 roc_auc = metrics.auc(fpr,tpr)
roc_auc
plt.plot(fpr,tpr)
plt.xlim([0,1])
plt.ylim([0,1])
plt.style.use("dark_background")
from sklearn.tree import export_graphviz
export_graphviz(dt, out_file ='tree.dot',
feature_names = ["Gender","Age","Salary"], class_names = ['0','1'],
rounded = True, proportion = False, precision = 2, filled = True)
!pip install pydotplus
!conda install graphviz
!dot tree.dot -Tpng -o image.png
```

OUTPUT:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
User ID             False
Gender              False
Age                 False
EstimatedSal
ary         False
Purchased
False   dtype:
bool
```

```
  array([['Male', 19,
           19000],
 ['Male', 35, 20000],
           ['Female', 26, 43000],
           ...,
           ['Female', 50, 20000],
           ['Male', 36, 33000],
           ['Female', 49, 36000]], dtype=object)
```

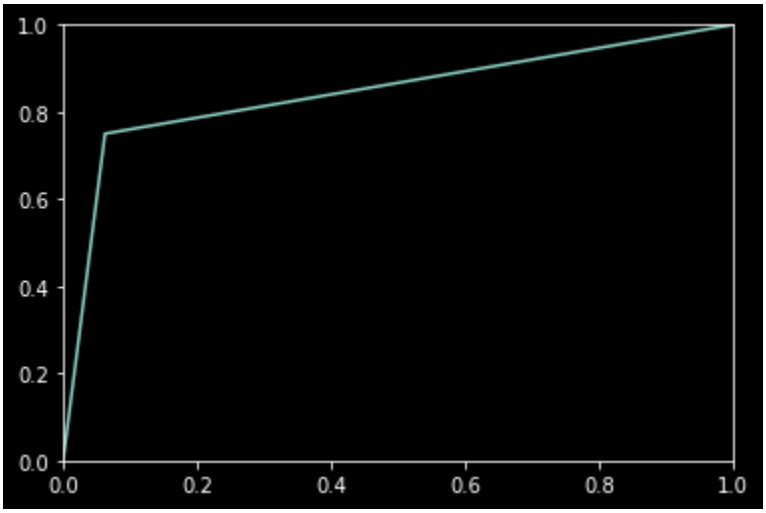
```
array([[0],
[0],
[0],
[0],
```

```
[0],
[0],
[0],
[1],
[0],
[0]], dtype=int64)


    array([[1,      19,   19000],
           [1,    35,   20000],
           [0,    26,   43000],
           ...,
           [0,    50,   20000],
           [1,    36,   33000],
           [0,    49,   36000]], dtype=object)


     array([[1,      27,   88000],
            [1,  41,   52000],
           [0,    27,   84000],
           ...,
           [1,    36,   52000],
            [0,  27,   54000],
            [0,  26,    118000]],  dtype=object)


    array([[1,      30,   87000],
           [0,  38,   50000],
           [1,  35,   75000],
           [0,  30,   79000],
           [0,  35,   50000],
           [1,  27,   20000],
           [0,  31,   15000],
           [1,  36,   144000],
           [0,  18,   68000],
           [1,  47,   43000]], dtype=object)

    DecisionTreeClassifier(class_weight=None, criterion='entropy',
    max_depth=N one,
                max_features=None, max_leaf_nodes=None,
                min_impurity_decrease=0.0, min_impurity_split=None,
                min_samples_leaf=1, min_samples_split=2,
                min_weight_fraction_leaf=0.0, presort=False,
                random_state=Non
    e,
                splitter='best')


     array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,                0,   0,  1,  0,   0, 1, 0, 0, 1,
            0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,             0,   0,  0,  0,   1], dtype=int64)
```

0.9

```
array([[30,  2],
       [ 2,  6]], dtype=int64)
```

0.84375



RESULT:

Thus the classification using decision classification using python is implemented.

EXERCISE NO:5 IMPLEMENT CLASSIFICATION USING PYTHON FOREST

DATE:11/08/2020 CLASSIFICATION

AIM:

To implement classification using random forest classification by using python.

DESCRIPTION:

Random forest, the name implies, consists of a large number of individuals decision tree that operate as an ensemble.

Each individual tree in the random forest splits out a class prediction and the class with the most votes become our model's prediction.

The reason for this wonderful effect is that the tree predict each other from their individual efforts.

While some tree may be wrong, many other tree will be right. So as group of tree are able to move in correct decision.

Random Forest Classification

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv("ads.csv")
dataset.head()
dataset.isnull().any()
x = dataset.iloc[:, 1:4].values y = dataset.iloc[:, 4].values
x[:5]
y[:5]
from sklearn.preprocessing import LabelEncoder lb=LabelEncoder() x[:,0]=lb.fit_transform(x[:,0])
x[:5]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.1, random_state = 0
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=10000,criterion='entropy')
rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score: ",accuracy_score(y_test,y_pred)*100,"%")
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(y_test,y_pred),columns=["Prediction -0","Prediction -1"],inde
import sklearn.metrics as metrics
fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred) roc_auc = metrics.auc(fpr, tpr)
print("AUC:",roc_auc)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

OUTPUT :

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
User ID                 False
Gender                  False
Age                     False
EstimatedSal
ary        False
Purchased
False    dtype:
bool
```

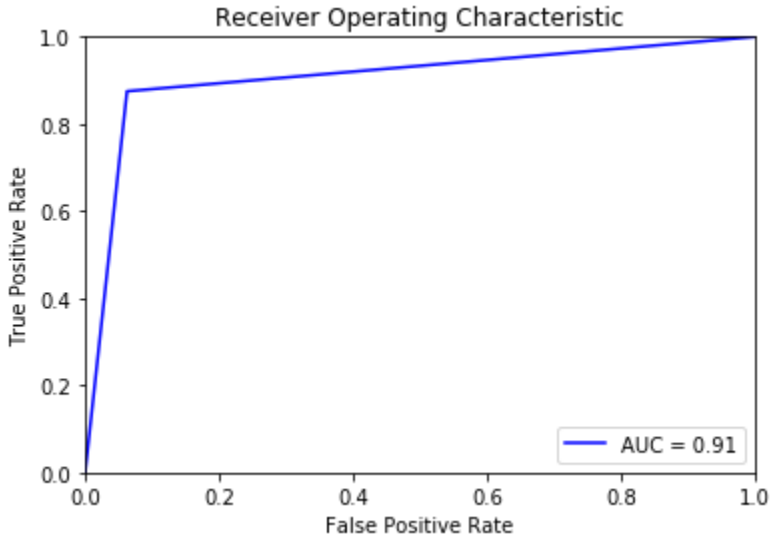
```
array([['Male', 19,
        19000],
       ['Male', 35,
        20000],
       ['Female', 26, 43000],
       ['Female', 27, 57000],
       ['Male', 19, 76000]], dtype=object)
```

```
array([0, 0, 0, 0, 0], dtype=int64)
```

```
array([[1,  19,  19000],
       [1,  35,  20000],
       [0,  26,  43000],
       [0,  27,  57000],
       [1,  19,  76000]], dtype=object)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entrop y',

one,

                       max_depth=None, max_features='auto', max_leaf_nodes=N

                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10000, n_jobs=None,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False)
```

	Prediction - 0	Prediction - 1
Actual - 0	30	2
Actual - 1	1	7



RESULT:

Thus the classification using python forest classification is implemented.

EXERCISE NO:6 **IMPLEMENT DECISION TREE BASED ALGORITHM FOR**

DATE:27/08/2020 **CLASSIFICATION**

AIM:

To implement decision tree based algorithm for classification using python.

DESCRIPTION:

Decision tree is a flow- chart where the structure where an internal note represents feature.

Partition is done on the basis of the attribute value.

In partition the tree in recursively manner called recursive partition.

This flow- chart like structure helps you in decision making.

Its visualization like a flow chart diagram which easily minimises the human level thinking.

Decision tree is easy to understand and interpret.

Decision Tree

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("ads.csv")
data.head()
data.isnull().any()
x = data.iloc[:,1:4].values y = data.iloc[:,4:5].values
x
y[:10]
from sklearn.preprocessing import LabelEncoder lb = LabelEncoder()
x[:,0] = lb.fit_transform(x[:,0])
 x
from sklearn.model_selection import train_test_split as tts x_train,x_test,y_train,y_test = tts(x,
y, test_size = 0.1,random_state=0)
x_train
x_test[:10]
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='entropy')
dt.fit(x_train,y_train)
y_pred = dt.predict(x_test)
y_pred
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
import sklearn.metrics as metrics
fpr, tpr ,threshold = metrics.roc_curve(y_test,y_pred)
roc_auc = metrics.auc(fpr,tpr)
roc_auc
plt.plot(fpr,tpr)
plt.xlim([0,1])
plt.ylim([0,1])
plt.style.use("dark_background")
from sklearn.tree import export_graphviz
export_graphviz(dt, out_file ='tree.dot',
feature_names = ["Gender","Age","Salary"], class_names = ['0','1'],
rounded = True, proportion = False, precision = 2, filled = True)
!pip install pydotplus
!conda install graphviz
!dot tree.dot -Tpng -o image.png
```

OUTPUT:

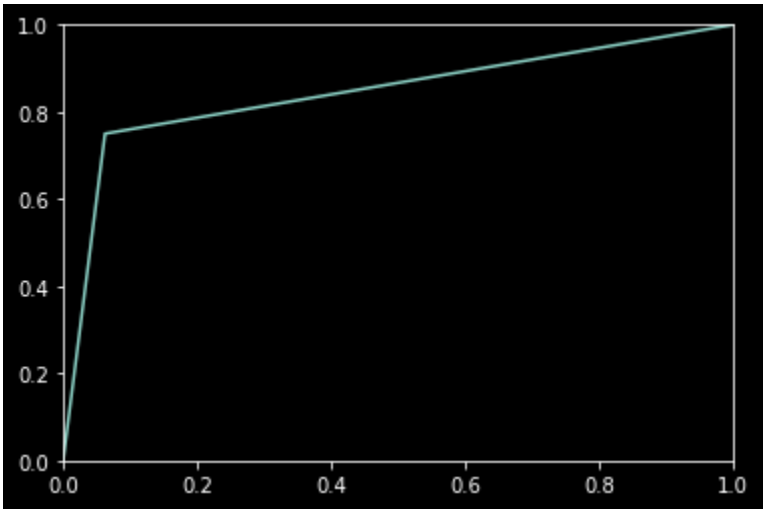
	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
User ID              False
Gender               False
Age                  False
EstimatedSal
ary        False
Purchased
False    dtype:
bool
```

```
  array([['Male', 19,
           19000],
 ['Male', 35, 20000],
           ['Female', 26, 43000],
           ...,
           ['Female', 50, 20000],
           ['Male', 36, 33000],
           ['Female', 49, 36000]], dtype=object)
array([[0],

[0],

[0],

[0],

[0],

[0],

[0],
```

```
[1],

[0],

[0]], dtype=int64)



array([[1,      19,   19000],
       [1,   35,   20000],
       [0,   26,   43000],
       ...,
       [0,   50,   20000],
       [1,   36,   33000],
       [0,   49,   36000]], dtype=object)




array([[1,      27,   88000],
       [1,  41,   52000],
       [0,   27,   84000],
       ...,
       [1,   36,   52000],
       [0,  27,   54000],
       [0,  26,    118000]],  dtype=object)

array([[1,      30,   87000],
       [0,  38,   50000],
       [1,  35,   75000],
       [0,  30,   79000],
       [0,  35,   50000],
       [1,  27,   20000],
       [0,  31,   15000],
       [1,  36,   144000],
       [0,  18,   68000],
       [1,  47,   43000]], dtype=object)

DecisionTreeClassifier(class_weight=None, criterion='entropy',
max_depth=N one,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=Non
e,
                       splitter='best')
```

```
    array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
           0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
           0,  0,  1,  0,  0, 1, 0, 0, 1,
           0,  0,  0,  0,  1], dtype=int64)
0.9

array([[30,  2],
    [ 2,  6]], dtype=int64)
0.84375
```



RESULT:

Thus the decision tree based algorithm for classification using python is implemented.

EXERCISE NO:7 BACK PROPAGATION NEURAL NETWORK ALGORITHM

DATE:08/09/2020

AIM:

To implement back propagation neural network algorithm using python.

DESCRIPTION:

The principle of back propagation approach is model a queen function by modifying internal weighting of input signals to produce an expected output signal.

The system is trained using a supervised learning method where the error between the system output and a known expected output is presented to the system and to modify the internal state.

Technically, this algorithm is a method for training the weights in a multilayer feed-forward neural networks.

Artificial Neural Network

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Churn_Modelling.csv') dataset.head()
dataset.isnull().any()
x = dataset.iloc[:, 3:13].values y = dataset.iloc[:, 13].values
x[:5]
y[:5]
from sklearn.preprocessing import LabelEncoder, OneHotEncoder labelencoder_X_1 = LabelEncoder()
x[:, 1] = labelencoder_X_1.fit_transform(x[:, 1]) labelencoder_X_2 = LabelEncoder()
x[:, 2] = labelencoder_X_2.fit_transform(x[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [1])
 x = onehotencoder.fit_transform(x).toarray()
x = x[:, 1:]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
from keras.models import Sequential
from keras.layers import Dense
model=Sequential()
model.add(Dense(input_dim=11,init="random_uniform",activation="relu",output_dim=20))
model.add(Dense(init="random_uniform",activation="relu",output_dim=15))
model.add(Dense(init="random_uniform",activation="sigmoid",output_dim=1))
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
model.fit(x_train,y_train,batch_size=32,epochs=100)
y_pred=model.predict(x_test)
y_pred
y_pred=y_pred>0.5
y_pred
from sklearn.metrics import accuracy_score
print("Accuracy score",accuracy_score(y_test,y_pred)*100,"%")
```

OUTPUT:

Number	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumO
1	15634602	Hargrave	619	France	Female	42	2	0.00	
2	15647311	Hill	608	Spain	Femal	41	1	83807.86	

					e			
3	15619304	Onio	502	France	Female	42	8	159660.80
4	15701354	Boni	699	France	Female	39	1	0.00
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82

```
RowNumber        False
CustomerId       False
Surname          False
CreditScore      False
Geography        False
Gender           False
Age              False
Tenure           False
Balance          False
NumOfProduct     False
HasCrCard        False
IsActiveMember   False
EstimatedSalary  False
Exited           False
dtype: bool
```

```
array([[619, 'France', 'Female', 42, 2, 0.0, 1, 1, 1, 101348.88],
       [608, 'Spain', 'Female', 41, 1, 83807.86, 1, 0, 1, 112542.58],
       [502, 'France', 'Female', 42, 8, 159660.8, 3, 1, 0, 113931.57],
       [699, 'France', 'Female', 39, 1, 0.0, 2, 0, 0, 93826.63],
       [850, 'Spain', 'Female', 43, 2, 125510.82, 1, 1, 1, 79084.1]],
      dtype=object)
```

```
array([1, 0, 1, 0, 0], dtype=int64)
```

```
array( [ [ 0.25752968 ] ,
         [ 0.36637256 ] ,
         [ 0.18552992 ] ,
         ...,
         [ 0.19875047 ] ,
         [ 0.17600316 ] ,
         [ 0.26788092 ] ] ,  dtype=float32 )
```

```
array( [ [ False ] ,
         [ False ] ,
```

```
        [ False ] ,
        ...,
        [ False ] ,
        [ False ] ,
        [ False ] ] )
```

Accuracy score 85.85000000000001 %

RESULT:

Thus the back propagation neural network using python is implemented.

EXERCISE NO:8 IMPLEMENT K-MEANS ALGORITHM FOR CLUSTERING

DATE:17/09/2020

AIM:

To implement k- means algorithm for clustering using python.

DESCRIPTION:

This procedure follows a simple and easy way to classify a given set through a certain number of clusters.

The centre should be placed in a cunning way because of different location causes different rules.

The better choice is to place them as much as possible far away from each other.

The next step is to take each point belonging to a given dataset and associate to the necessary centre.

If no dataset point was recognized then stop.

K-Means algorithm

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Mall_Customers.csv') dataset.head()
dataset.isnull().any()
x = dataset.iloc[:,[3,4]].values
x[:5]
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
   kmeans = KMeans(n_clusters = i,init = 'k-means++' , max_iter = 300, n_init =
   10,random_st kmeans.fit(x)
   wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
kmeans = KMeans(n_clusters=5,init = 'k-means++' , max_iter = 300, n_init=10,random_state=0
y_kmeans = kmeans.fit_predict(x)
y_kmeans[:5]
plt.scatter(x[y_kmeans == 0,0],x[y_kmeans == 0,1],s=100,c='red',label='cluster 1') plt.scatter(x[y_kmeans
== 1,0],x[y_kmeans == 1,1],s=100,c='blue',label='cluster 2') plt.scatter(x[y_kmeans == 2,0],x[y_kmeans
== 2,1],s=100,c='green',label='cluster 3') plt.scatter(x[y_kmeans == 3,0],x[y_kmeans ==
3,1],s=100,c='yellow',label='cluster 4') plt.scatter(x[y_kmeans == 4,0],x[y_kmeans ==
4,1],s=100,c='brown',label='cluster 5')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='black',label
plt.title('Clusters of clients')
plt.xlabel("Annual Income in 1000 $") plt.ylabel("Spending Score (1-1000") plt.legend()
plt.show()
```

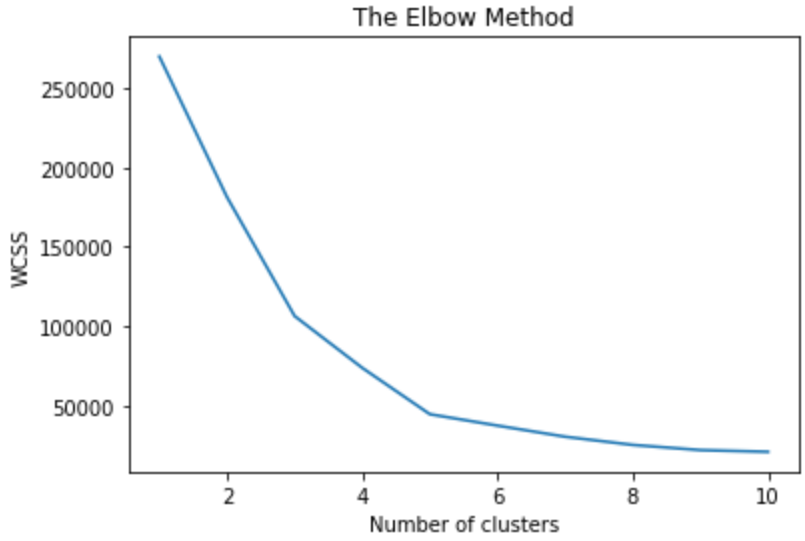
OUTPUT:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Femal	31	17	40

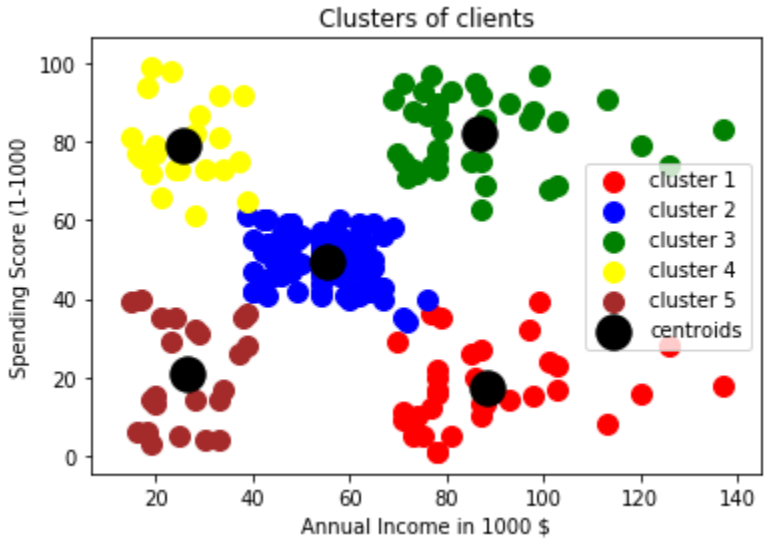
e

```
CustomerID     False
Genre  False
Age      False
Annual Income (k$) False
Spending Score (1-100) False
dtype: bool
```

```
array([[15,     39],
       [15,  81],
       [16,    6],
       [16,  77],
       [17,  40]], dtype=int64)
```




```
array([4, 3, 4, 3, 4])
```


Clusters of clients
Spending Score (1-1000
Annual Income in 1000 $
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
centroids
0
20
40
60
80
100
20
40
60
80
100
120
140

RESULT:

Thus the k-means algorithm for classification using python is implemented.

EXERCISE NO:9 IMPLEMENT HIERARCHICAL CLUSTERING ALGORITHM

DATE:06/10/2020

AIM:

To implement hierarchical algorithm for cluster the dataset.

DESCRIPTION:

As the name suggests it is an algorithm that builds hierarchy of clusters.

The algorithm starts with all the data point assigned to the cluster of their own.

Then two nearest cluster are merged into same clusters.

In the end, this algorithm terminates when there is a single cluster left.

The decision of merging two clusters is taken on the basis of closeness of these clusters.

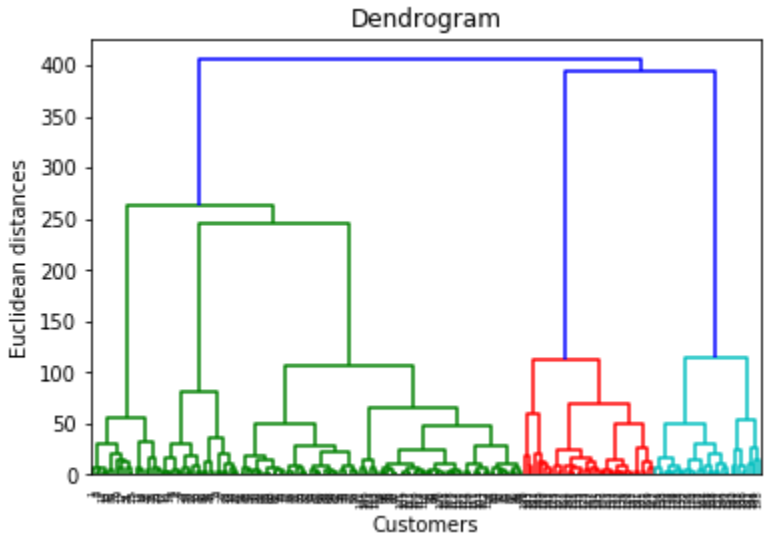
Hierarchical Clustering

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Mall_Customers.csv') x = dataset.iloc[:, [3, 4]].values
x[:5]
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(x, method = 'ward')) plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
 y_hc = hc.fit_predict(x)
```

```
plt.scatter(x[y_hc == 0, 0] x[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(x[y_hc == 1, 0] x[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(x[y_hc == 2, 0] x[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(x[y_hc == 3, 0] x[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(x[y_hc == 4, 0] x[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)') plt.ylabel('Spending Score (1-100)') plt.legend()
plt.show()
```

OUTPUT:

```
array([[15,  39],
       [15,  81],
       [16,   6],
       [16,  77],
       [17,  40]],  dtype=int64)
```



array([4, 3, 4, 3, 4], dtype=int64)

RESULT:

Thus the hierarchical clustering algorithm is implemented.

EXERCISE NO:10 IMPLEMENT APRIORI ALGORITHM FOR ASSOCIATE

Date:22/10/2020 RULE

AIM:

To implement apriori algorithm for associate rule.

DESCRIPTION:

This algorithm is used to gain insight into the structure relationships between different items involved.

And also used for mining frequent itemsets and relevant association rules.

It assumes that any subset of a frequent item set must be frequent.

Association rule mining is a technique to identify frequent patterns and association among a set of items.

Apriori Algorithm

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Market_Basket_Optimisation.csv',header = None) dataset.head()
transactions = []
for i in range(0, 7501):
transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])
transactions[:2]
from apyori import apriori
rules = apriori(transactions, min_support = 0.003, min_confidence = 0.2, min_lift = 3, min_
results = list(rules)
results
```

OUTPUT:

	0	1	2	3	4	5	6	7	8	9	10
0	shrimp	almonds	avocado	Vegetables mix	Green grapes	Whole wheat flour	yams	Cottage cheese	energy drink	tomato juice	Low fat yogurt
1	burgers	meatballs	Eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Turkey	Avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Mineral water	milk	Energy bar	Whole wheat rice	Green tea	NaN	NaN	NaN	NaN	NaN	NaN

[['shrimp', 'almonds', 'avocado', 'vegetables mix', 'green grapes', 'whole weat flour', 'yams',

'cottage cheese', 'energy drink', 'tomato juice', 'low fat yogurt', 'green tea', 'honey',

'salad', 'mineral water', 'salmon',

'antioxydant juice', 'frozen smoothie',

[RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.004

532728969470737,
ordered_statistics=[OrderedStatistic(items_base=frozenset ({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829 059829057, lift=4.84395061728395)]),
RelationRecord(items=frozenset({'mushroom cream sauce', 'escalope'}), sup port=0.005732568990801226,
ordered_statistics=[OrderedStatistic(items_base =frozenset({'mushroom cream sauce'}), items_add=frozenset({'escalope'}), c onfidence=0.3006993006993007, lift=3.790832696715049)]),
RelationRecord(items=frozenset({'escalope', 'pasta'}), support=0.00586588 4548726837,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'pa sta'}), items_add=frozenset({'escalope'}), confidence=0.3728813559322034, lift=4.700811850163794)]),
RelationRecord(items=frozenset({'honey', 'fromage blanc'}), support=0.003 332888948140248,
ordered_statistics=[OrderedStatistic(items_base=frozenset ({'fromage blanc'}), items_add=frozenset({'honey'}), confidence=0.24509803 92156863, lift=5.164270764485569)]),
RelationRecord(items=frozenset({'herb & pepper', 'ground beef'}), support=0.015997866951073192, ordered
statistics=[OrderedStatistic(items base=fro

RESULT:

Thus the apriori algorithm for associate rule is implemented.